BSc(Hons) Computing Science
Proposals for Student Projects

2015/16
Contents

Mitigating ‘Virtual Host Confusion Attacks’ on TLS
Mark Vella ..................................................... 1

Hardening Windows Network Authentication
Mark Vella ..................................................... 3

Securing Android against Capability Leaks
Mark Vella ..................................................... 5

Real-time object recognition in videos with a parallel algorithm
George Azzopardi ........................................... 7

Implementation and Comparison of Data Embedding Algorithms
Johann A. Briffa ............................................. 8

High Density 2D Barcodes - Research
Johann A. Briffa ............................................. 9

High Density 2D Barcodes - Implementation
Johann A. Briffa ............................................. 10

Information Hiding using Imperceptible Yellow Printer Dots
Johann A. Briffa ............................................. 11

Updates to Distributed Simulator for Communications Systems
Johann A. Briffa ............................................. 13

Image Forensics
Johann A. Briffa ............................................. 14

Telemetry-based optimisation for user training in racing simulators.
Keith Bugeja, Sandro Spina ............................... 15

Real-time radiosity for dynamic scenes.
Keith Bugeja, Kevin Vella ............................... 16

Cloud-based Rendering of Natural Video on Mobile Devices
Carl James Debono ......................................... 18

Heart Beat Rate Calculation from Facial Video Recording on Smartphones
Carl James Debono ......................................... 19

Fast Video Compression
Carl James Debono ......................................... 20

Multi-hop Data Broadcasting in Vehicular Networks
Carl James Debono ......................................... 21

An LLVM Bitcode Interpreter for 16-bit MSP430 Microcontrollers
Dr. Joshua Ellul ............................................. 22

A Domain Specific Language for Automated Testing of iOS Applications
Mark Micallef ................................................ 23

A Domain-Specific Language for Layman-Driven Hypothesis Testing
Gordon Pace, Mark Micallef ............................ 24

Using Contract Automata to Monitor Service Composition
Gordon Pace ................................................ 26
Adaptive Games using Runtime Monitoring
  
  Gordon Pace, Sandro Spina, Keith Bugeja .......................... 27

Cheat Detection for Games using Runtime Verification
  
  Gordon Pace, Sandro Spina, Keith Bugeja .......................... 29

Game Design using Monitor-Oriented Programming
  
  Gordon Pace, Sandro Spina ........................................... 31

Bridging LTL and QuickCheck
  
  Gordon Pace, Christian Colombo ..................................... 33

Runtime Verification of Javascript Programs
  
  Gordon Pace, Christian Colombo ..................................... 34

Interactive Exploration of Explanations of Formal Proofs
  
  Gordon Pace, Christian Colombo ..................................... 36

Asynchronous PollyRV
  
  Gordon Pace; Christian Colombo ...................................... 38

Monitoring of UML specifications
  
  Christian Colombo, Gordon Pace ...................................... 40

GUI and Feature Adaptation through Runtime Verification
  
  Gordon Pace; Christian Colombo ...................................... 41

Testing Properties using System-Side Mocking
  
  Gordon Pace, Christian Colombo ...................................... 42

A Parametrisable Mock Transaction System
  
  Gordon Pace, Christian Colombo ...................................... 44

Profiling Monitoring Overheads
  
  Gordon Pace, Christian Colombo ...................................... 46

An Eclipse Plugin for LARVA or polyLARVA
  
  Gordon Pace, Christian Colombo ...................................... 47

Static and Dynamic Property Management for LARVA or polyLARVA
  
  Gordon Pace, Christian Colombo ...................................... 48

Debugging Techniques for Runtime Verification
  
  Gordon Pace, Christian Colombo ...................................... 50

Communicating Sequential Processes with JavaScript ES6 Generators
  
  Kevin Vella ............................................................. 52

Visualising and Interacting with Music-Theoretic Structures
  
  Kevin Vella, Gordon Pace ............................................... 53

Towards Offline Usage of JavaScript Web Applications
  
  Joshua Ellul, Kevin Vella ............................................... 54

Multi-core Thread Scheduling
  
  Kevin Vella, Keith Bugeja, Sandro Spina ............................. 55

Supervisor Index .......................................................... 56

Keyword Index .................................................................. 58
Mitigating ‘Virtual Host Confusion Attacks’ on TLS

Supervised by: Mark Vella
Keywords: Network Security, Digital Forensics, Incident Response
Level: Undergraduate final year project

The Transport Layer Security (TLS) protocol is the de-facto standard for providing secure channels for web applications [4]. In theory, this approach can mitigate network attackers positioned as a man-in-the-middle (MiTM). However, as with all security protocols in general, the devil is in the (implementation) detail. In their recent BlackHat talk, Delignat-Lavaud and Bhargavan [1] reported that all formal guarantees for TLS make the strong assumption that a physical server host is always associated with a single IP address and corresponding single domain name. However in the real-world, large-scale sites make use of Content Delivery Networks (CDN) [5] in order to scale to bandwidth requirements that completely invalidates this assumption due to the critical usage of certificate-sharing virtual hosts. The onset of cloud computing is giving rise to a similar situation. The primary consequence is the circumvention of the Same-Origin-Policy (SOP) implemented by web browsers [3], amplifying the impact of the ever-relevant cross-site scripting (XSS) attacks [2].

This project aims to mitigate this threat by:

- Conducting a risk assessment in terms of pre-requisites and impact for this threat. This task requires implementing the attack idea within a penetration testing tool and executing it against simulated vulnerable configurations.

- Identifying associated forensic artifacts in order to enable for successful responses in case of successful exploitation.

- Providing a hot-patch for web-servers, using just-in-time binary modification, as part of an instantaneous intrusion recovery procedure.

Bibliography


Hardening Windows Network Authentication

Supervised by: Mark Vella
Keywords: Network Security, Digital Forensics, Incident Response
Level: Undergraduate final year project

Windows domain controller make use of the Kerberos protocol in order to provide network authentication and consequently proper access control within local area networks\(^1\). This is considered an important step forward from the vulnerable challenge-response based NTLM authentication used for workgroup configurations [2]. However, as with any Key Distribution Center (KDC)-based authentication [3], Windows domain controllers present a single point-of-failure for the entire network’s security. The way these controllers are typically configured by default through the windows administration management console, though, further aggravates this known weakness as it increases the likelihood of exploitation. Moreover, an intruder attempting lateral movement from a compromised workstation has other avenues to attain this goal without even having to compromise a domain controller [1]. Given the widespread use of windows server technology for deploying organizational networks, this situation is of considerable concern.

This project aims to harden such deployments by:

- Conducting a risk assessment in terms of pre-requisites and impact for this threat. This task requires implementing the reported weaknesses as part of a penetration testing tool and executing it against simulated vulnerable setups.
- Identifying associated forensic artifacts in order to enable for successful responses in case of successful exploitation.
- Providing a patch for domain controllers and workstations-alike, using just-in-time binary modification, for longer-term network hardening.

Bibliography


\(^1\)https://msdn.microsoft.com/en-us/library/bb742516.aspx#EBAA
Securing Android against Capability Leaks

Supervised by: Mark Vella  
**Keywords:** Systems Security, Digital Forensics, Incident Response  
**Level:** Undergraduate final year project

The Android operating system has by far retained its majority market share of stock smartphones over the past years\(^3\). While indicative of its success, its wide-spread adoption also renders Android-based devices prone to abuse by malware campaigns. Specifically the same powerful framework components that enable Android applications to interoperate seamlessly - e.g. intents, service and content provider application components\(^4\) - pose the primary threat. These features could in fact be abused by malware in a manner to even bypass Android’s permissions system\(^5\). Existing work has mainly focused on detecting weaknesses inside applications whose security-critical functionality could be exposed to malicious apps \([2, 4, 1, 3]\). Approaches that attempt to secure the Android OS directly exist \([5]\), however these rely on information flow tracking that could be difficult to get right in terms of complete monitoring.

This project aims to secure the Android OS against this threat by:

- Conducting a risk assessment in terms of pre-requisites and impact. This task requires implementing proof-of-concept malware as part of a penetration testing tool.
- Identifying associated forensic artifacts in order to enable for successful detection in case of successful infections.
- Providing an Android patch, using just-in-time binary modification, that focuses on firewalling the trust boundaries between application components.

**Bibliography**

\(^3\)http://www.idc.com/prodserv/smartphone-os-market-share.jsp  
\(^4\)http://developer.android.com/guide/components/fundamentals.html  


Real-time object recognition in videos with a parallel algorithm

Supervised by: George Azzoppardi
Keywords: Real-Time, Parallel, Object Recognition, Video Analysis
Level: Undergraduate final year project

Real-time object recognition in videos is an ever growing field with various applications ranging from video surveillance to robotics, among others. In this project the selected candidate will have the opportunity to evaluate the effectiveness of a recently published method applied in real-time video processing. The student will be required to implement the algorithm in parallel mode, possibly on a GPU, and perform experiments on a benchmark data set.

The concerned algorithm is called COSFIRE which stands for Combination of Shifted Filter Responses, and has already been demonstrated to be highly effective in various applications, including traffic sign detection and recognition in complex scenes, feature detection in retinal images, and image classification. For further details I refer to the publication [1] and to the Matlab implementation which can be downloaded from: http://de.mathworks.com/matlabcentral/fileexchange/37395

Bibliography

Implementation and Comparison of Data Embedding Algorithms

Supervised by: Johann A. Briffa

Keywords: Level: Undergraduate final year project

Spatial-domain information hiding in images consists in the modification of pixel data in order to represent some message sequence. The resistance of each embedding method to lossy image compression schemes (such as JPEG) and the detectability of the embedding have a direct impact on the usability of these schemes.

Objectives: Investigate the relationship between the parameters for the hidden information (such as message length, embedding strength, etc.) and the resistance to image compression and detectability

Student background/interests:

• Interest in information hiding.

• Some knowledge of digital imaging fundamentals.

• Aptitude and willingness to program (the project is likely to use a selection of languages, including C++ and Python).

Bibliography
High Density 2D Barcodes - Research

Supervised by: Johann A. Briffa
Keywords: 
Level: Undergraduate final year project

A number of standards for 2D barcodes exist, which allow the encoding of more data (in the same physical space) than conventional 1D barcodes. Various companies, including Microsoft and HP, have shown interest in techniques that increase the encoding density—that is, the amount of data that can be encoded per unit area. Other researchers have also used techniques from watermarking and steganography to create barcodes that do not look random.

Objectives: Investigate existing standards and proposed techniques, comparing their embedding density and decoding reliability. It is expected that some of these techniques will need to be implemented and tested.

Student background/interests:

- Interest in information hiding.
- Some knowledge of digital imaging fundamentals.
- Aptitude and willingness to program (the project is likely to use a selection of languages, including C++ and Python).

Bibliography


High Density 2D Barcodes - Implementation

Supervised by: Johann A. Briffa

Keywords:

Level: Undergraduate final year project

A number of standards for 2D barcodes exist, which allow the encoding of more data (in the same physical space) than conventional 1D barcodes. Various companies, including Microsoft and HP, have shown interest in techniques that increase the encoding density—that is, the amount of data that can be encoded per unit area. Other researchers have also used techniques from watermarking and steganography to create barcodes that do not look random.

Objectives: Investigate and compare existing standards and proposed techniques. Implement a reader for a specific system on an embedded or portable device (such as a mobile phone).

Student background/interests:

- Interest in information hiding.
- Some knowledge of digital imaging fundamentals.
- Aptitude and willingness to program.
- Knowledge of Java and Android development would be helpful.

Bibliography
Information Hiding using Imperceptible Yellow Printer Dots

Supervised by: Johann A. Briffa
Keywords:       Level: Undergraduate final year project

Since the introduction of high quality colour laser printers, the US Government has required colour laser printouts to imperceptibly embed document tracking information. This has been achieved by creating an imperceptible grid of tiny yellow dots across the page. Yellow dots are used to take advantage of a weakness in the human visual system that makes seeing yellow on white difficult. If the dots need to be extracted, a blue light can be shone on page, resulting in the dots becoming visible, and thus enabling manual detection. Our previous work has demonstrated that the same technique can be used to embed arbitrary data, which can then be automatically extracted after printing and scanning.

Objectives: This project will look to continue our previous work in this area. There are two possible streams that could be looked at.

1. Optimising the placement of the yellow dots in order to both reduce errors and increase capacity. This may also involve looking at whether it is possible to interleave our dots with those printed by the laser printer itself currently we print our documents on ink jet printers that do not print their own yellow dots. This is a more practical based approach and will involve conducting extensive testing and require good programming skills to implement any changes.

2. A more theoretical approach would look at optimising the error correction code currently used. We initially used a naive error correction code as proof of concept, however, there is substantial scope for optimisation. This would involve both conducting tests to evaluate the characteristics of the channel (printing and scanning documents to determine error rates) as well as constructing a more optimised error correction code.

Student background/interests:
- Interest in information hiding
- Willingness to program the current implementation is in Java, so good Java skills are essential
- For stream 2 a background in mathematics or error coding is desirable
Bibliography
Updates to Distributed Simulator for Communications Systems

Supervised by: Johann A. Briffa
Keywords: 
Level: Undergraduate final year project

SimCommSys is a multi-platform distributed Monte Carlo simulator for communication systems. The error control coding component implements various kinds of binary and non-binary codes, including turbo, LDPC, repeat-accumulate, and Reed-Solomon. This code base has been in continuous development since 1997, and currently weighs in at over 45,700 physical lines of code, written by Dr. Briffa and collaborators. The distributed computing component of this code uses a client/server architecture built on TCP/IP to facilitate running simulations on grid resources; this also works well on local clusters.

Objectives: This project will look to extend the existing code base, continuing our previous work in this area. Various extensions could be looked at, including:

- Writing a cross-platform GUI for the simulator (i.e. writing software to create and edit simulation files in a user-friendly way).
- Writing a back-end / middle-ware for matching resources with simulations.
- Adding a result validation component to confirm simulation reproducibility and facilitate the use of public computing.

Student background/interests:

- Interest in low-level computing issues and parallel computing.
- Aptitude and willingness to program (the project requires the use of C++; prior OO development experience in another language is suitable).

Bibliography
Image Forensics

Supervised by: Johann A. Briffa
Keywords:
Level: Undergraduate final year project

This topic involves participation in the supervisor’s ongoing work in multimedia security. The topic of image forensics deals with issues related to the authentication of digital images, or the identification of their provenance. Usually these topics are of interest to cases in the courts of law, where such images are used as evidence.

Objectives: Depends on identified problems within the supervisor’s ongoing work, as agreed with the student.

Student background/interests:
• Interest in image forensics / multimedia security.
• Some knowledge of digital imaging fundamentals.
• Aptitude and willingness to program (the project is likely to use a selection of languages, including C++ and Python).

Bibliography
Telemetry-based optimisation for user training in racing simulators.

Supervised by: Keith Bugeja, Sandro Spina
Keywords: Games, Optimisation
Level: Undergraduate final year project

Serious games are computer games whose main purpose is to train or educate users [3]. In a number of cases, a serious game is designed to facilitate the acquisition or exercise of different skills within a particular domain. In the case of driving-related serious games, driving simulators embellished with adequately tailored exercises have been used for a variety of purposes, including teaching of basic driving skills [1] and the analysis of driver fatigue [2]. This FYP investigates the design of a serious game intended to refine user driving skills by providing continuous feedback about driving behaviour. In particular, the student is expected to explore methods for analyzing telemetry from car instrumentation and devise a quasi-real-time feedback mechanism to drive user improvement. A user study will be carried out to assess the effectiveness of the feedback system against a control group.

Bibliography


Real-time radiosity for dynamic scenes.

Supervised by: Keith Bugeja, Kevin Vella  
Keywords: Computer Graphics, Ray Tracing, GPU  
Level: Undergraduate final year project

The radiosity algorithm [3] is a technique for the estimation of exitant radiance in an environment, a view-independent method which describes the amount of illumination leaving one surface and reaching another using the finite element method. The technique is principally used in the context of diffuse reflectors. These surfaces are approximated via a finite number of planar patches with constant radiosity and reflectivity; a form factor denotes the fraction of energy leaving one patch and arriving at another. In order to compute the form-factor for a patch, the visibility between the patch and all patches over the hemisphere of directions above the patch must be determined. This is typically carried out via projection techniques [4], computationally expensive operations which constrain surfaces to static configurations in real-time scenarios.

In this project, we will investigate a novel algorithm which employs GPU programmable shaders to compute a voxel representation of the scene. Subsequently, ray-casting [1, 2, 5] and spatial-hashing methods will be used to estimate the form-factors used in the radiosity solution. The goal of this project that of applying the radiosity method to dynamic scenes in real-time.

Bibliography


Cloud-based Rendering of Natural Video on Mobile Devices

Supervised by: Carl James Debono
Keywords: Video Processing, Cloud Computing, Computer Graphics
Level: Undergraduate final year project

Nowadays, mobile devices are being equipped with more processing power and graphical processing units allowing them to perform more complex tasks. Free-viewpoint television is a solution that allows the user to select a view where this can be an actual camera view or a virtual position that has to be generated from the available views. Rendering these virtual views at 25 or 30 frames per second or better for high-definition screens is very computationally demanding even for the latest devices and drains battery power fast. A potential solution to mitigate this workload is to offload part of these computations to the cloud. The aim of this project is to develop and evaluate a fast cloud-based rendering scheme that can be deployed in the mobile environment.

Bibliography
Heart Beat Rate Calculation from Facial Video Recording on Smartphones

Supervised by: Carl James Debono
Keywords: Video Processing, Digital Image Processing, Artificial Intelligence
Level: Undergraduate final year project

Mobile devices have become portable computers which are equipped with cameras, colour displays, and other sensors. A non-invasive technique to measure heart beat is to use video processing. Image processing tools are used to determine the location of the face of the user as the area of interest. This is followed by video processing algorithm to identify and count the number of heart beats per minute. This can be displayed on the screen or transmitted as necessary. Given that the video is taken in real-life scenarios, motion induced artifacts and ambient noise have to be filtered. The aim of this project is to create a solution that is low in complexity, fast, and reliable with the resources available on a typical smartphone.

Bibliography
Fast Video Compression

Supervised by: Carl James Debono  
Keywords: Video Processing, Computer Graphics  
Level: Undergraduate final year project

The latest video compression standard (H.265 or HEVC) is quite complex due to the amount of predictions and searches required in optimising compression. This results in delays that make it unpractical in resource-restricted environments. Therefore there is scope to try to minimise time complexity by employing better prediction schemes and parallelisation of the algorithms used. The aim of this project is therefore to explore techniques and develop algorithms to minimise the time taken for compression. The study will include the impact of these algorithms on the quality of the video.

Bibliography
Multi-hop Data
Broadcasting in Vehicular
Networks

Supervised by: Carl James Debono
Keywords: Data Networks, Graph Theory, Artificial Intelligence
Level: Undergraduate final year project

New vehicles are being equipped with on-board computers and soon will carry wireless devices that can transmit and receive data. For example, it is common today to have a GPS receiver on-board a vehicle. Such wireless devices will have the possibility to communicate with other vehicles and with the infrastructure with the aim of improving road safety. Broadcasting is ideal for vehicular networks since fast dissemination of critical information is crucial for its effectiveness. However, the reliability of transmission cannot be guaranteed in such an environment. The aim of this project is to develop a multi-hop broadcasting protocol and investigate its performance compared to simple flooding of the network.

Bibliography
An LLVM Bitcode Interpreter for 16-bit MSP430 Microcontrollers

Supervised by: Dr. Joshua Ellul
Keywords: Virtual Machines, Operating Systems, Compilers, C, Assembly Code
Level: Undergraduate final year project

Over the past decade LLVM [3] has been gaining more traction and adoption by both academia and the industry alike. LLVM provides a compiler infrastructure and intermediate representation (IR) that allows different source languages (e.g. C and Java) to be compiled for different architectures. LLVM uses its IR as the target of the first 'Frontend' compilation pass. A final 'Backend' compilation pass is performed on IR to produce the native code for the target architecture (e.g. x86, ARM, etc). IR is often smaller than native code. In resource constrained microcontrollers having tens of kilobytes of program space and around ten kilobytes of memory, space is of prime importance.

In this project we will investigate techniques to interpret LLVM IR code on resource constrained 16-bit microcontrollers in an efficient manner. We will aim to specifically implement an interpreter for MSP430 microcontrollers, however will also focus on facilitating other resource constrained microcontrollers. In doing so we will investigate intermediate representation design [1], garbage collection and memory management [2] techniques.

Bibliography


A Domain Specific Language for Automated Testing of iOS Applications

Supervised by: Mark Micallef
Keywords: Testing, Test Automation
Level: Undergraduate final year project

Test automation has become a mainstream activity when it comes to ensuring that software is functioning properly. However, whereas technology in domains such as web applications has become standardised and well-known, significant challenges remain when it comes to automating mobile applications.

A Domain Specific Language (DSL) is defined as a computer programming language of limited expressiveness focused on a particular domain. A well-designed DSL enables users of the language to express notions in that domain in a concise, understandable and unambiguous manner. In this FYP, we aim to investigate the possibility of creating a DSL which will allow test engineers to express tests for the iOS domain and subsequently compile those tests into an executable test suite which can be executed in an iOS device or simulator. The project consists of both a theoretical language design component and a technical component. The language design component involves studying the iOS domain, extracting a taxonomy of concepts and subsequently designing a language. The technical component involves the compilation of a subset of the designed language into automated tests.

Bibliography
A Domain-Specific Language for Layman-Driven Hypothesis Testing

Supervised by: Gordon Pace, Mark Micallef  
Keywords: Domain Specific Languages  
Level: Undergraduate final year project  
Level: MSc

In contrast with general-purpose languages, which allow for the specification of any algorithm, but invariably at a cost in the complexity of designing systems and debugging them, domain-specific languages (DSLs) [1] are typically syntactically constrained to be used in a single domain — limiting their expressivity, but allowing them to be used by non-technical persons. Commonly used DSLs include HTML for the description of the structure of web content, VHDL for hardware description and SQL for database access and manipulation. The advantage of using such languages is that they typically provide (i) frequently used units of computation for the domain (e.g. a drawing DSL might provide a primitive to rotate or skew an image); and (ii) means of ‘glueing’ together different units of computation in a manner which is, once again, particular to the domain (e.g. a hardware description language might have means of putting two circuits in series or in parallel).

Systems developed using such languages, thus tend to be readable by domain-experts without also necessitating technical expertise and, with appropriate support (IDEs, debugging, user-feedback, etc) also allow them to write such systems themselves. This avoids the need to communicate domain specific issues to technical (but not domain) experts and short-circuits a feedback loop in the development of such systems when the feature requested by the domain expert does not match the expected result (is it a bug in the implementation or was the specification flawed in the first place?)

We have recently started a project with the HR department of an international (but locally-based) company, who is seeking to be able to manipulate employee data to answer questions and develop hypotheses they might have (e.g. ‘Do workers who do more overtime take more sick leave?’). In this project, the aim is to build a domain specific language to support the processing of tabular data posing of such questions, giving sufficient feedback to be able to identify issues which can be explored more concretely by statisticians and data scientists. The project will include the use of real-life data from this company.
and the involvement of management personnel to allow for the evaluation of the accessibility of the DSL.

Bibliography

Using Contract Automata to Monitor Service Composition

Supervised by: Gordon Pace
Keywords: Contracts, Runtime Verification
Level: MSc

Level: Undergraduate final year project

Contract automata [3] present a way of expressing the expected behaviour of interacting systems. For instance, a contract automaton may state that the client is permitted to initialise the service, after which he or she is obliged to submit valid user credentials.

One way of enforcing such a contract is to set up an intermediate node acting as a proxy between the two parties, and ensuring that no communication between the parties violates the ideal behaviour specified in the contract. Runtime verification [2] tools such as polyLARVA [1] focus precisely on ensuring that a system conforms to a given behaviour at runtime.

The aim of this project is to implement a proxy for service invocation, which enforces a given contract using polyLARVA.

Bibliography


Adaptive Games using Runtime Monitoring

Supervised by: Gordon Pace, Sandro Spina, Keith Bugeja
Keywords: Runtime Verification, Games
Level: MSc

Level: Undergraduate final year project

The gaming industry is a fast growing lucrative industry with many challenges. Game players have come to expect high-quality games which revolve around their needs. Developing games is challenging in many respects: a game has to be user friendly, fast responding, interesting and challenging but not disheartening, fair and usually visually appealing.

A challenging aspect of games is that it is virtually impossible to test a game for user experience quality without having human players actually playing. Therefore, over and above the complex game logic, a game would require various probes for measuring user experience and any other relevant information. This information can then be used as feedback to the game, making it adapt according to user feedback. For example, if a user seems more interested in the strategy aspect of the game, then the game should provide more strategy problems to keep the player involved.

Runtime verification [1] provides an approach to monitoring the behaviour of a system without directly changing the code — thus separating the concerns of normal system behaviour from that of the verification. An application of this technique is monitor-oriented programming [2], in which the system behaviour is actually modified through feedback from the monitors e.g. whenever the monitor identifies that a user is requesting a file, credentials are requested to ensure that the user has the necessary rights.

The aim of this project is to apply a monitor-oriented approach to games, enabling easier measuring of user experience and adaptation of user experience. Without changing the game logic, monitoring can be used to harvest rich information concerning user experience during the game and use it to give feedback to the game to support adaptation according to user preferences.

To test the approach, it will be implemented over an existing game and evaluated for cheat detection.

Bibliography

Cheat Detection for Games using Runtime Verification

**Supervised by:** Gordon Pace, Sandro Spina, Keith Bugeja  
**Keywords:** Runtime Verification, Games  
**Level:** Undergraduate final year project

Cheating in games has long been a serious problem, threatening the trust of players in the fairness of the game. Cheats are particularly problematic when human players play against each other and even more if players pay for playing the game. One such example is World of Warcraft where players pay considerable sums of money to play. Cheats have direct effects in the user experience and the number of players. Therefore cheat detection is taken very seriously by game developers. For instance, Blizzard Entertainment use Warden, which monitors the game behaviour and the operating system for potential cheats.

The main problem with cheat detection is that it is very hard to emulate all real-life situations which might lead to cheating under laboratory conditions. This necessitates a runtime approach to cheat detection instead of an a priori approach of trying to detect all possible cheating situations. An applicable technique is runtime verification [1], which supports runtime monitoring of a system’s behaviour, comparing it to the ideal behaviour, taking appropriate action whenever a violation occurs.

In this project, we propose a monitoring architecture for games which enables easier measuring of user behaviour and cheat detection. Without changing the game logic, monitoring can be used to harvest rich information during the game and use either game specific rules or machine learning techniques to identify abnormal behaviour which may be indicative of cheating. The proposed architecture, shown below, uses monitoring of a game, collecting information and forwarding it to a results gather and processing module. At this stage, the data is quite raw and would require processing to deduce useful information such as the occurrence of a cheat.

![Cheat Detection Architecture Diagram](image_url)
Bibliography

Game Design using Monitor-Oriented Programming

Supervised by: Gordon Pace, Sandro Spina
Keywords: Runtime Verification
Level: Undergraduate final year project

Runtime verification [2] provides an approach to monitoring the behaviour of a system without directly changing the code — thus separating the concerns of normal system behaviour from that of the verification. An application of this technique is monitor-oriented programming [3], in which the system behaviour is actually modified through feedback from the monitors e.g. whenever the monitor identifies that a user is requesting a file, credentials are requested to ensure that the user has the necessary rights.

The aim of this project is to apply a monitor-oriented approach to design and build a Space Invaders style game based on a library for handling input and output, and using polyLarva [1]. The game logic can be implemented as rules in the runtime monitoring tool. For instance, upon receiving an event triggered by the player pressing the left direction key the spaceship $x$-coordinate is decremented by sending an event to the screen system, which automatically triggers another rule which changes the display accordingly. Similarly, when the collision detection algorithm triggers an overlap between a bullet and the spaceship, the user’s life count is decremented, while a rule which restarts the game triggers when the life count becomes zero.

By structuring the rules appropriately, the project will aim to assess how well adapted monitor-oriented programming using rule-based specifications are for the development and customisation of games.

Bibliography


Bridging LTL and QuickCheck

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification
Level: Undergraduate final year project

Level: MSc

QuickCheck is a tool which automatically generates tests to test whether a system adheres to a particular property. Currently QuickCheck does not support property specification in LTL — a popular language used to specify trace properties, e.g., “It is always the case that the alarm sounds until the fire is put out”.

There are two main challenges: first, LTL is defined in terms of infinite traces while QuickCheck generates finite traces; second, LTL properties are more abstract than the code and thus links have to be created to bridge the gap.

The aim of this project is to explore these challenges and build a library that uses LTL as a high level language on top of QuickCheck.

Bibliography
Runtime Verification of Javascript Programs

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification
Level: Undergraduate final year project

Level: MSc

Runtime verification techniques [1, 5] have been successfully employed in a number of areas (e.g. [3]) and tool support for various platforms has also been on the rise [4]. PollyRV [2] (originally named polyLarva) is an extensible runtime verification framework which can be extended through plugins to support multiple technologies simultaneously, and language plugins already exist for Java, C, Erlang and PHP.

This project aims to build a stand-alone monitoring framework, or extend PollyRV with support for Javascript scripting language as the implementation language for both the client-side logic, and server-side system, with particular focus on the interplay between the two sides. The system will be evaluated using an open-source ecommerce backend, primarily evaluating (i) whether business-logic properties can be effectively monitored; and (ii) the overheads induced on such a system.

Bibliography


Interactive Exploration of Explanations of Formal Proofs

Supervised by: Gordon Pace, Christian Colombo
Keywords: E Learning
Level: Undergraduate final year project

Upon first encountering formal proofs, students usually require time and practice to understand the syntactic nature of formal reasoning. After years of seeing (at best) rigorous proofs, restraining oneself from taking ‘obvious’ shortcuts (mistakes in a formal setting) takes time to get used to. And yet, the structure of formal proofs makes their mechanical checking trivial to do. Furthermore, reading a proof from a book from top to bottom does not help the reader understand how he or she would have derived the proof in the first place. Consider the following formal proof of the contrapositive law, taken from [1]:

1. \( \neg (\neg Q \implies \neg P) \) (sub-hypothesis)
2. \( P \) (sub-hypothesis)
3. \( P \implies Q \) (hypothesis)
4. \( Q \) (= \(-\)elimination on lines 3 and 2)
5. \( \neg Q \implies \neg P \) (lemma \( A \vdash \neg A \implies B \) on line 4)

6. \( P \) (sub-hypothesis)
7. \( \neg (\neg Q \implies \neg P) \) (copy line 1)
8. \( \neg P \) (= \(-\)introduction on subproofs 2–5 and 6–7)
9. \( \neg Q \implies \neg P \) (lemma \( A \vdash B \implies A \) on line 8)

10. \( \neg (\neg Q \implies \neg P) \) (sub-hypothesis)
11. \( \neg \neg (\neg Q \implies \neg P) \) (= \(-\)introduction on subproofs 1–9 and 10–10)
12. \( \neg Q \implies \neg P \) (= \(-\)elimination on line 11)

Understanding how the proof writer started by assuming \( \neg (\neg Q \implies \neg P) \) is not obvious, in particular because the proof writer would have actually started by trying to prove the result by negation elimination on the last line.

In this project, the aim is to provide students a formal means of writing propositional and predicate logic proofs with an interactive tool which can (i) allow the user to write a proof which the program will check for correctness; (ii) attempt to automatically derive a proof for formulae in a subset of the logic; and (iii) given a proof, explain how one would have constructed it, in a step-by-step manner with explanatory text in between. The tool will allow new students to interactively learn about formal proofs and get natural language explanations following the order of construction of the proof.
Bibliography

Asynchronous PollyRV

**Supervised by:** Gordon Pace; Christian Colombo  
**Keywords:** Runtime Verification  
**Level:** Undergraduate final year project

Runtime verification techniques [1, 6] have been successfully employed in a number of areas (e.g., [4]) and tool support for various platforms has also been on the rise [5]. PollyRV [2] (originally named polyLarva) is an extensible runtime verification framework which can be extended through plugins to support different technologies. In this project, we propose to exploit the plugin architecture to enable asynchronous runtime verification. The advantage of this approach is that the system is not slowed down since the monitor would not need to keep up with the stream of system events. The aim of the project will be that of investigating different approaches to building a plugin to be used with PollyRV to enable asynchronous monitoring of systems, and evaluate the results, (also comparing them to online monitoring) using the tool SMock [3].

**Bibliography**


Monitoring of UML specifications

Supervised by: Christian Colombo, Gordon Pace
Keywords: Runtime Verification
Level: Undergraduate final year project


Although UML statecharts are regularly used to specify ideal system behaviour, neither of the tools support UML as a specification language. This project aims to write a translator from UML properties for both tools and compares their efficiency through a case study. A major challenge will be that of allowing feedback from the runtime verification tools to the person writing properties using UML statecharts through this same notation, rather than through the tool’s native notation.

Bibliography


GUI and Feature Adaptation through Runtime Verification

Supervised by: Gordon Pace; Christian Colombo  
Keywords: Runtime Verification  
Level: Undergraduate final year project

Runtime verification techniques [1, 3] have been successfully employed in a number of areas (e.g. [2]) particularly where it is difficult to predict how a piece of software will be used once deployed. One such example is a non-trivial GUI which typically offers multiple ways of achieving the same functionality.

Using runtime verification feature and GUI usage patterns can be observed and classified (e.g. the usage pattern of an expert user would be significantly different from that of a beginner), enabling the possibility of automatic interface adaptation for future users (e.g. hiding advanced features for beginners). This project will investigate these ideas through the use of a case study.

Bibliography


Testing Properties using System-Side Mocking

Supervised by: Gordon Pace, Christian Colombo
Keywords: Embedded Systems, Runtime Verification, Testing
Level: MSc

Level: Undergraduate final year project

Although techniques such as runtime verification [2] are used to guarantee that a system adheres to a specification, as specification grow in size and complexity, the question which naturally arises is whether the specification is really the one had in mind, and really embodies the notion of correctness. The aim of this project is to enable the exploration and testing of a specification to ensure that it is a correct one.

Most runtime verification techniques, such as Larva [1], support specifications over events occurring on the system-side — for instance, a specification may state that initialise should occur before getFile, and that no more than 100 instances of getFile may occur in any one minute period. The aim of this project is to create a tool to enable:

Animation of a specification through the automated generation of a dashboard for a user to experiment with. The tool would automatically generate such a dashboard through which the specification writer may fire events through the use of buttons, and allow time to advance through the interface.

Testing of the specification through the generation of an API which is then used in conjunction with a testing tool to generate different input over which to test the specification. Essentially, the approach would be mocking the system to enable unit testing of the specification. Another application of this interface would be to measure the overheads induced by the monitors.

The approach would be evaluated through the development of a large specification case study, which will be tested using the environment.

Bibliography

A Parametrisable Mock Transaction System

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification
Level: MSc
Level: Undergraduate final year project

Like any piece of software, runtime verification tools which generate monitoring code from formal specification have to be adequately tested, particularly so because of its use to assure other software. State-of-the-art runtime verification tools such as Java-MOP [6] and tracematches [2] have been tested on the DaCapo benchmark [1]. However, the kind of properties which have been monitored are rather low level contrasting with our experience with industrial partners who seem more interested in checking for higher level properties (such as the ones presented in [4, 3]). Whilst we had the chance to test our tool Larva [5] on industrial case studies, such case studies are usually available for small periods of time and in limited ways due to privacy concerns. Relying solely on such case studies can be detrimental for the development of new systems which need substantial testing and analysis before being of any use.

For this reason, we have developed a testing framework which provides a highly configurable mock transaction system to enable thorough validation of systems which interact with it. This project aims to extend this project in two ways:

- The testing framework is based on a scripting language to enable the user to configure the mock. A possible direction of the project involves the formalisation of the semantics of this scripting language.

- A number of advanced features of the testing framework have not yet been implemented. Another potential avenue of the project involves the implementation of these features.

Bibliography

[2] Eric Bodden, Laurie J. Hendren, Patrick Lam, Ondrej Lhoták, and No- 
mair A. Naeem. Collaborative runtime verification with tracematches. J. 

aware runtime monitoring. In Runtime Verification - First International 
Conference, (RV), volume 6418 of Lecture Notes in Computer Science, pages 

based runtime monitoring of real-time and contextual properties. In Formal 
Methods for Industrial Critical Systems (FMICS), volume 5596 of Lecture 

monitoring of real-time java programs (tool paper). In Seventh IEEE Inter-
national Conference on Software Engineering and Formal Methods (SEFM), 

[6] Patrick O’Neil Meredith, Dongyun Jin, Dennis Griffith, Feng Chen, and 
Grigore Roşu. An overview of the MOP runtime verification framework. 
To appear; http://dx.doi.org/10.1007/s10009-011-0198-6.
Profiling Monitoring Overheads

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification
Level: Undergraduate final year project

Monitoring systems for correctness comes at a price — the monitors consume memory and take time to execute, thus slowing down the system and robbing it of valuable resources. This has been one of the major drawbacks of runtime verification [3] when applied to real-life systems, especially ones for which performance is crucial, or ones that run in an environment with reduced resources.

Typically, profiling of the monitoring overhead requires running the system with and without the monitors on identical inputs in identical environments, a process which should ideally be automated to avoid slowing down the testing process, especially in a rapid development environment. Being able to measure overheads easily also helps property exploration and optimisation (sometimes a property may be changed in such a manner that it uses less resources, but still captures the same original property).

One approach to enable efficient profiling of monitors is that of explicitly tagging the parts of the code introduced for monitoring, and using a profiling tool to extract information about the resources consumed by this code with respect to the overall system. The aim of this project is to extend the tool LARVA [2] or polyLARVA [1] to enable the use of a profiler in the code it generates.

Bibliography


An Eclipse Plugin for Larva or polyLarva

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification
Level: Undergraduate final year project
Level: MSc

Larva [2] and polyLarva [1] are two runtime verification tools which, given a system and a specification produce an altered system with its behaviour monitored in such a manner that any violation of the specification is flagged, and the execution potentially stopped. The aim of this project is to write a plugin for Eclipse to (i) support the development and debugging of specification scripts; and (ii) include these scripts as parts of a project, supporting compilation with and without monitors, and possibly link the specification to the system when debugging.

The main aim of the project will be to explore the integration of runtime verification as part of a standard IDE, evaluating the result through its use in the development of a case study.

Bibliography


Static and Dynamic Property Management for Larva or polyLarva

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification
Level: Undergraduate final year project
Level: MSc

The use of validation and verification techniques for software systems increases the confidence in the system mainly due to the fact that properties and specification typically function at a more abstract level than the actual software — normally addressing when a particular behaviour is correct, rather than how to compute the correct behaviour e.g. it is far easier to write a property which checks that a given timetable contains no clashes, than to write a program which computes a timetable with no clashes.

However, as systems grow in size and complexity, so do their specifications. Software developers using tools to check the system being constructed against specifications require tools to manage the specifications in the same way they require tools to manage the software system being written.

In particular, runtime verification [3] techniques, which enable installation of monitors which operate at runtime to check the system for conformance, usually require the developer of the specification to experiment with different (possibly equivalent) properties to reduce overheads on the deployed system.

The aim of this project is to explore the development of property management techniques and tools for the Larva [2] or polyLarva [1] runtime verification tools.

Property specifications: Since the runtime verification tools Larva and polyLarva support different logics, a requirements engineer would ideally have access to a way of describing and defining properties in terms of different languages, and combine them together e.g. one may write a property $P$ as a regular expression, and another $P'$ using LTL and then define the aggregate property which states that at least one of the two properties should hold at any point in time. Furthermore, a requirements engineer should be able to manage properties and tag them appropriately, for instance splitting a requirement into a number of smaller ones, or keeping different versions of the same property (possibly to be used in different environments).

Static property management: The project will aim at developing a plugin
for Eclipse to enable the manage specifications and also decide, at compile
time, which properties are to be woven into the system.

**Dynamic property management:** A possible extension to the project (or a
separate project) would be to explore the extension of the property
management to be managed dynamically at runtime, enabling for instance
properties to be switched on or off, and their parameters changed. For
instance, one may decide to switch off monitoring of a property when the
system is under stress, or may decide to change the threshold value used to
classify a transaction as potentially fraudulent at runtime without having
to restart the system.

**Bibliography**

Extensible technology-agnostic runtime verification. In Barbora Buhnova,
Lucia Happe, and Jan Kofroň, editors, Proceedings 10th International Workshop
on *Formal Engineering Approaches to Software Components and Architectures*,
2013.

[2] Christian Colombo, Gordon J. Pace, and Gerardo Schneider. Dynamic event-
based runtime monitoring of real-time and contextual properties. In *Formal
Methods for Industrial Critical Systems (FMICS)*, volume 5596 of *Lecture

tion. *Journal of Logic and Algebraic Programming*, 78(5):293–303, may/june
2009.
Debugging Techniques for Runtime Verification

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification
Level: MSc

Level: Undergraduate final year project

Runtime verification [3] enables the monitoring of a system at execution time, comparing it against a given specification for violations in an attempt to stop or remedy unexpected behaviour. Whenever a violation is discovered, reparatory action can be taken as required (and as specified).

However, violations typically indicate a malfunctioning system, and can usually be traced back to errors in the software, meaning that unless the system is fixed, the problem will probably arise again and again. To ensure that developers can address the bugs discovered, it is crucial that the runtime verification tools store and report sufficient information to enable them to understand what went wrong, and possibly replay the relevant behaviour that led to the bug.

The aim of this project is to enhance either the runtime verification tool Larva [2] or polyLarva [1] to allow for (i) stepwise debugging of system and specifications in an IDE; (ii) whenever a bug is discovered, report sufficient information to enable replaying the system up to the point of violation; (iii) extend the replaying capabilities to include the specification, allowing for better understanding of what went wrong.

Bibliography


Communicating Sequential Processes with JavaScript
ES6 Generators

Supervised by: Kevin Vella
Keywords: Concurrency, Distribution, Systems, JavaScript
Level: Undergraduate final year project

The sixth iteration of JavaScript (JavaScript 6, ECMAScript 6 or ES6), is arriving imminently [1]. It brings with it a number of powerful new features such as generators, which effectively enable the descheduling and rescheduling of event handlers in execution. This provides an opportunity to provide co-scheduling (cooperative scheduling) capabilities through a new JavaScript library, whereby executing entities can block on a synchronisation object and resume execution later, on rendezvous. This greatly facilitates the processing of asynchronous events by avoiding 'callback hell', since the programmer can instead rely on multiple threads of control to harness asynchronicity.

Communicating Sequential Processes (CSP) [2], introduced by Tony Hoare in the 1970s, is a formalism for expressing interactions in concurrent systems. CSP is a cornerstone of various concurrent programming languages, such as occam and Go, and can also be exposed as a library (Clojure’s core.async library and Java’s JCSP). Typically, CSP implementations use named synchronous channels for message passing between processes or threads.

This project centres around the provision of CSP channels in JavaScript by leveraging the power of ES6 generators. Existing work such as JS-CSP has already made inroads in this area, and the student is expected to familiarise his or herself with these developments before tackling the main problem of implementing CSP channels over a variety of transport mechanisms (local shared memory, web sockets, web worker messages and so forth). Ultimately, the aim is to allow cooperatively scheduled threads both on the server side (multiple node.js instances) and on the client side (browsers with multiple web workers) to communicate and synchronise independently of their location.

Bibliography


Visualising and Interacting with Music-Theoretic Structures

Supervised by: Kevin Vella, Gordon Pace
Keywords: Visualisation, Music Theory, Gamification
Level: Undergraduate final year project

Musical scales and chords, subsets of the spectrum of notes at a composer’s disposal, are an important fixture of music theory. Although one may dismiss the choice of such subsets as a purely aesthetic consideration, a growing body of work seeks to establish links with mathematical notions in group theory and euclidean geometry. Intriguingly, the most important musical scales turn out to be solutions to optimisation problems.

We believe there is an opportunity to develop innovative software tools for the interactive visualisation and exploration of music-theoretic structures [1]. The objective of this final year project is to design and implement an interactive application which enables the user to visualise pitch class sets (chords and scales) and their temporal behaviour (chord and key changes) using a variety of representations. A central issue is the use of interactivity to enable the user to build harmonic scenarios and investigate the effect of, for instance, chord substitutions using both visual and auditory representations. A major aspect of this project will entail exploring opportunities for gamification, so that the game player would be able to solve visual puzzles while experiencing real-time auditory feedback that sonifies his or her progress.

Bibliography

Towards Offline Usage of JavaScript Web Applications

Supervised by: Joshua Ellul, Kevin Vella
Keywords: JavaScript, Distribution, Systems, Cloud Computing
Level: Undergraduate final year project

Internet access is ubiquitous. Office and home environments are starting to make use of cloud computing platforms to facilitate day to day tasks. The main challenges highlighted by academia include: server consolidation, energy management, data security, storage technologies and data management [4]. Most literature either avoids issues pertaining to Internet connectivity or assumes that Internet connectivity is available (and rightly so), and as pointed out by [3]: “No Internet, no cloud computing - it’s that simple.”. However, situations often arise whereby connectivity is limited or unavailable (such as whilst traveling) and access to the cloud is still required. Offering offline access to web-based applications often involves implementing a desktop version of the cloud based application.

In this project we aim to investigate, propose and implement techniques to facilitate a framework for building JavaScript web-based applications that allows for their usage even when offline (without having to implement a desktop-based version). We plan to achieve this by implementing language constructs that will define what parts of the code and data models can be used whilst offline. A client based web server infrastructure (such as node.js [1] on the Google V8 engine [2]) can then be used to provide offline access to web-based applications by synchronising the respective code and data models onto the client side automatically.

Bibliography


Multi-core Thread Scheduling

**Supervised by:** Kevin Vella, Keith Bugeja, Sandro Spina  
**Keywords:** Systems, Concurrency, Parallel Computing  
**Level:** Undergraduate final year project

The aim of this project is to investigate techniques for scheduling light-weight threads on contemporary multi-core processors, with the aim of minimising overheads and maximising the utilisation of resources. A number of experimental designs for thread schedulers will be proposed and implemented, and their relative merits and shortcomings compared and contrasted through empirical measurement. Issues to be considered include the impact of alternative scheduling queue arrangements, contention on shared scheduler data structures, effective utilisation of the cache hierarchy, automatic thread migration across cores and integration with the operating system.

Bibliography
Supervisor Index

Azzopardi, George, 7

Briffa, Johann A., 8–11, 13, 14
Bugeja, Keith, 15, 16, 27, 29, 55

Colombo, Christian, 33, 34, 36, 38, 40–42, 44, 46–48, 50

Debono, Carl James, 18–21

Ellul, Joshua, 54

Micallef, Mark, 23, 24

Pace, Gordon, 24, 26, 27, 29, 31, 33, 34, 36, 38, 40–42, 44, 46–48, 50, 53

Spina, Sandro, 15, 27, 29, 31, 55

Vella, Kevin, 52–55
Vella, Mark, 1, 3, 5
Keyword Index

Artificial Intelligence, 19, 21
Cloud Computing, 18, 54
Computer Graphics, 16, 18, 20
Concurrency, 52, 55
Contracts, 26
Data Networks, 21
Digital Forensics, 1, 3, 5
Digital Image Processing, 19
Distribution, 52, 54
Domain Specific Languages, 24
E Learning, 36
Embedded Systems, 42
Games, 15, 27, 29
Gamification, 53
GPU, 16
Graph Theory, 21
Incident Response, 1, 3, 5
JavaScript, 52, 54
Music Theory, 53
Network Security, 1, 3
Object Recognition, 7
Optimisation, 15
Parallel, 7
Parallel Computing, 55
Ray Tracing, 16
Real-Time, 7
Runtime Verification, 26, 27, 29, 31, 33, 34, 38, 40–42, 44, 46–48, 50
Systems, 52, 54, 55
Systems Security, 5
Testing, 23, 42
Video Analysis, 7
Video Processing, 18–20
Visualisation, 53